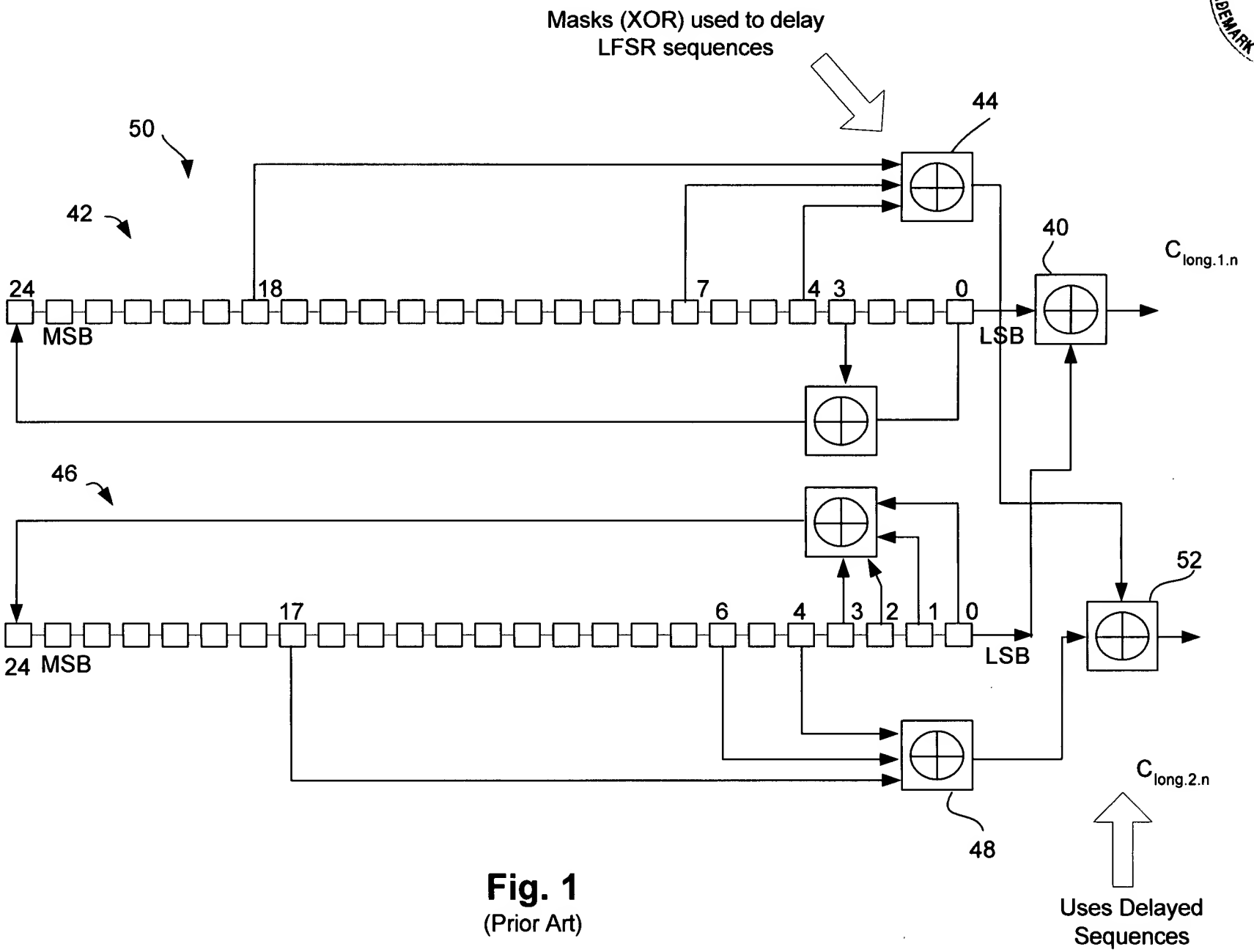

Fig. 1
(Prior Art)

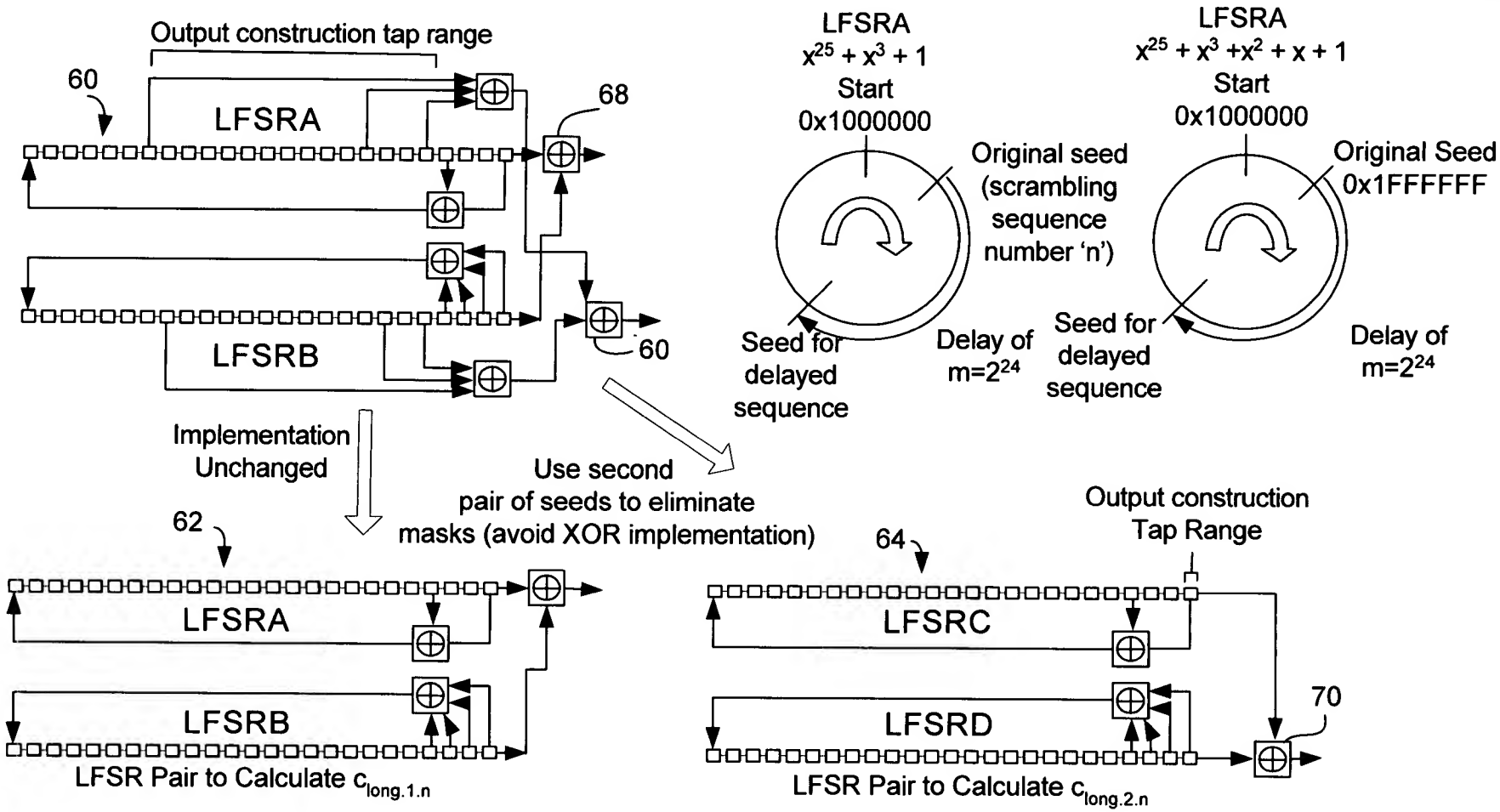

Fig. 2

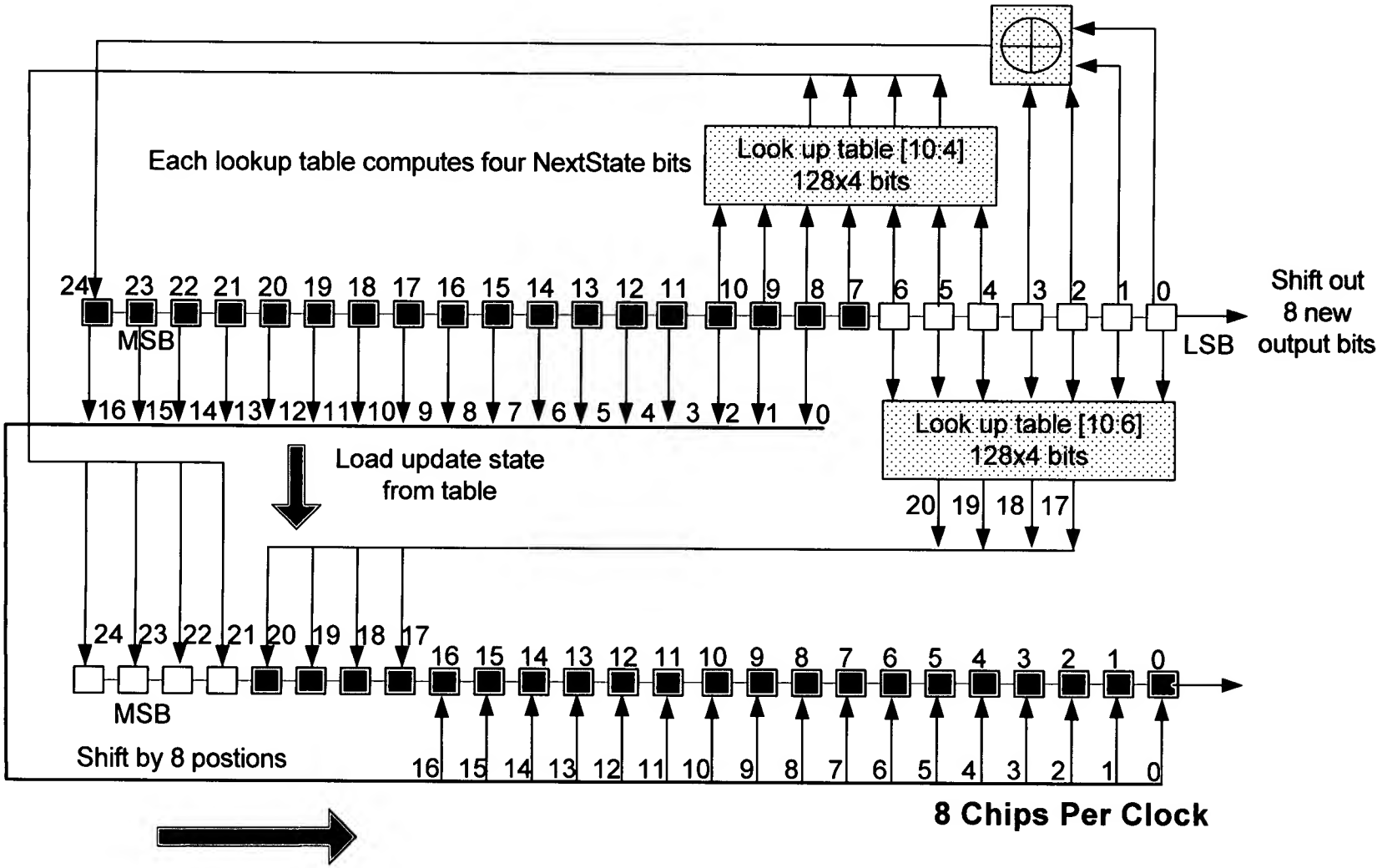

Fig. 3

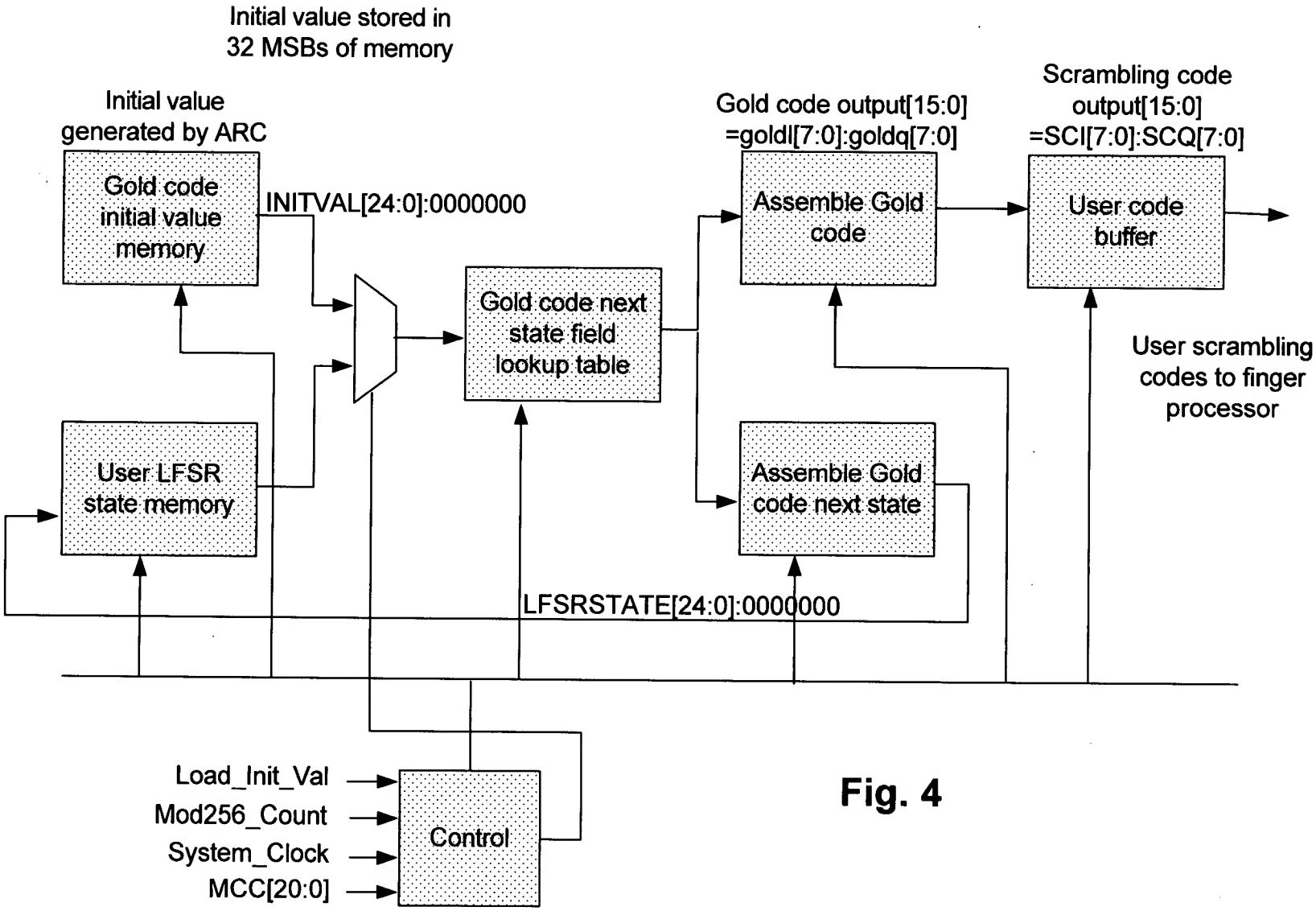

Fig. 4

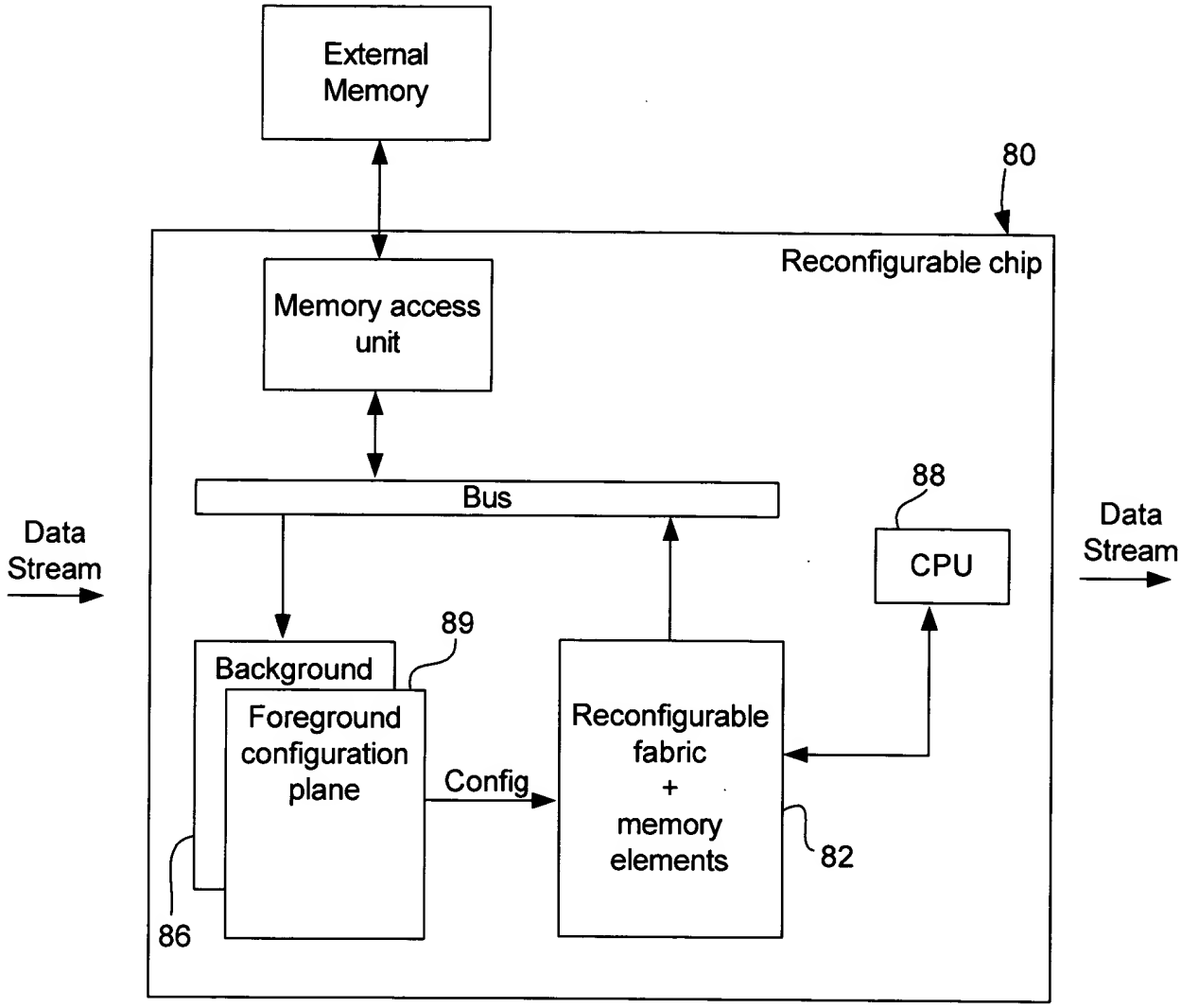

Fig. 5

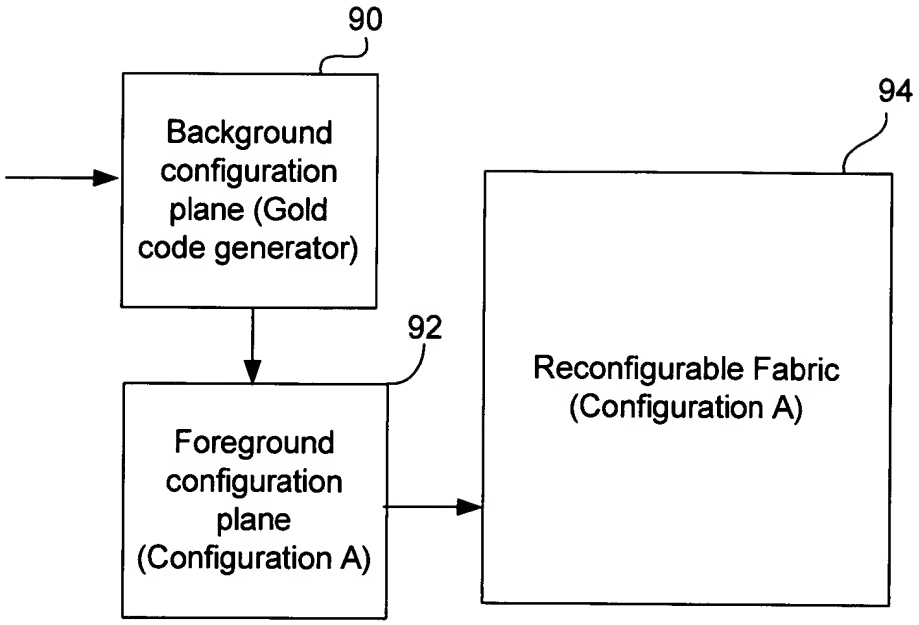

Fig. 6A

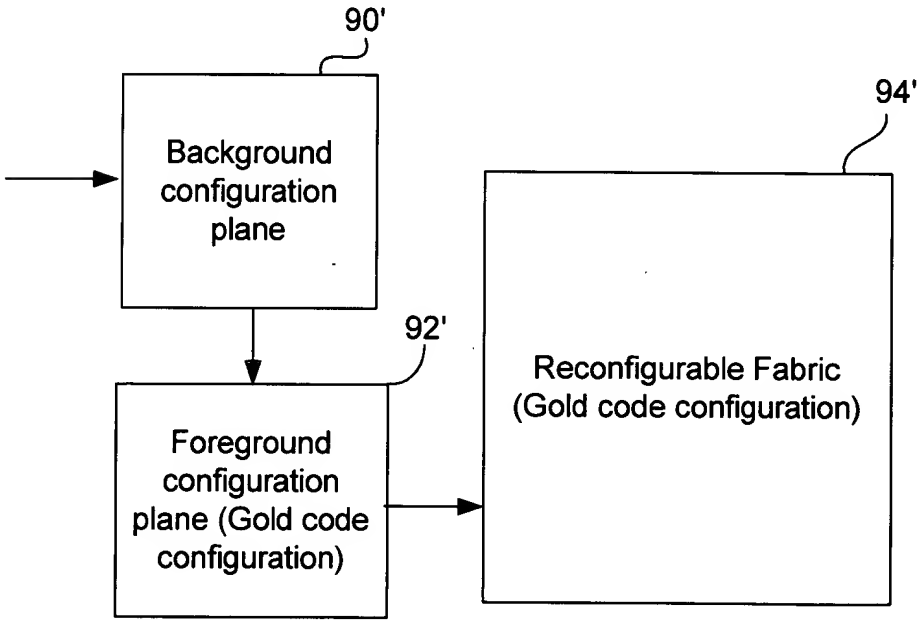

Fig. 6B

$C_{long1.n}$ = LFSRA[7:0] XOR LFSRB[7:0]
Let us define LFSRC'[i] = LFSRC[2[I/2]]

$C_{long.n}(i) = C_{long.n}(i)(1+j(-1)^{i}(c_{long2,}n(2[i/2])$ (From 3G TS25.213)
Multiplying bits by +1/-1 is the same as XOR for 0s and 1s.
XORing by 0xAA can be used in place of the $(-1)^{i}$ term.
In binary representation, the Scrambling code $C_{long.n}$ becomes:

$C_{long.n}[7:0] = C_{long1.n}[7:0](1+j(0xAA)$ XOR $C'_{long2.n}[7:0])$

$C_{long.n}[7:0]$ = LSFRA[7:0] XOR LFSRB[7:0]
+J(LFSRA[7:0] XOR LFSRB[7:0] XOR 0xAA XOR LFSRC'[7:0] XOR LFSRD'[7:0])

$C_{long.n}[7:0]$ = SCI[7:0] = Jscq[7:0]

Let us define LFSRD''[7:0] = 0xAA XOR LFSRD'[7:0], then:

$C_{long.n}[7:0]$ = (LFSRA[7:0] XOR LFSRB[7:0])
+ j(LFSRA[7:0] XOR LFSRB[7:0] XOR LFSRC'[7:0] XOR LFSRD''[7:0])

We use a lookup table to compute LFSRC'[7:0] and LFSRD''[7:0])

Fig. 7

Gold code generator lookup[6:0] definitions

At address 4n+0: OUT[7:0] = Next StateA[3:0]:PASSA[3:0] OUT[7] = IN[6] XOR IN[3] OUT[6] = IN[5] XOR IN[2] OUT[5] = IN[4] XOR IN[1] OUT[4] = IN[3] XOR IN[0] OUT[3] = IN[3] OUT[2] = IN[2] OUT[1] = IN[1] OUT[0] = IN[0]	At address 4n+2: OUT[7:0] = Next StateC[3:0]:LFSRC'[3:0] OUT[7] = IN[6] XOR IN[3] OUT[6] = IN[5] XOR IN[2] OUT[5] = IN[4] XOR IN[1] OUT[4] = IN[3] XOR IN[0] OUT[3] = IN[3] OUT[2] = IN[2] OUT[1] = IN[1] OUT[0] = IN[0]
At address 4n+1: OUT[7:0] = Next StateB[3:0]:PASSA[3:0] OUT[7] = IN[6] XOR IN[5] XOR IN[4] XOR IN[3] OUT[6] = IN[5] XOR IN[4] XOR IN[3] XOR IN[2] OUT[5] = IN[4] XOR IN[3] XOR IN[2] XOR IN[1] OUT[4] = IN[3] XOR IN[2] XOR IN[1] XOR IN[0] OUT[3] = IN[3] OUT[2] = IN[2] OUT[1] = IN[1] OUT[0] = IN[0]	At address 4n+3: OUT[7:0] = Next StateD[3:0]:LFSRD''[3:0] OUT[7] = IN[6] XOR IN[5] XOR IN[4] XOR IN[3] OUT[6] = IN[5] XOR IN[4] XOR IN[3] XOR IN[2] OUT[5] = IN[4] XOR IN[3] XOR IN[2] XOR IN[1] OUT[4] = IN[3] XOR IN[2] XOR IN[1] XOR IN[0] OUT[3] = /IN[2] OUT[2] = IN[2] OUT[1] = /IN[0] OUT[0] = IN[0]

Fig. 8A

Gold code generator lookup[10:4] definitions

At address 4n+0: OUT[7:0] = IN[7.4] Next StateA[7:4] OUT[7] = IN[3] OUT[6] = IN[2] OUT[5] = IN[1] OUT[4] = IN[0] OUT[3] = IN[6] XOR IN[3] OUT[2] = IN[5] XOR IN[2] OUT[1] = IN[4] XOR IN[1] OUT[0] = IN[3] XOR IN[0]	At address 4n+0: OUT[7:0] = IN[7:4] Next StateA[7:4] OUT[3] = IN[2] OUT[2] = IN[2] OUT[1] = IN[0] OUT[0] = IN[0] OUT[7] = IN[6] XOR IN[3] OUT[6] = IN[5] XOR IN[2] OUT[5] = IN[4] XOR IN[1] OUT[4] = IN[3] XOR IN[0]
At address 4n+1: OUT[7:0] = IN[7:4] Next StateB[7:4] OUT[7] = IN[6] OUT[6] = IN[5] OUT[5] = IN[4] OUT[4] = IN[3] OUT[3] = IN[3] XOR IN[5] XOR IN[4] XOR IN[3] OUT[2] = IN[2] XOR IN[4] XOR IN[3] XOR IN[2] OUT[1] = IN[1] XOR IN[3] XOR IN[2] XOR IN[1] OUT[0] = IN[0] XOR IN[2] XOR IN[1] XOR IN[0]	At address 4n+1: OUT[7:0] = IN"[7:4]Next StateB[7:4] OUT[3] = /IN[2] OUT[2] = IN[2] OUT[1] = /IN[0] OUT[0] = IN[0] OUT[7] = IN[6] XOR IN[5] XOR IN[4] XOR IN[3] OUT[6] = IN[5] XOR IN[4] XOR IN[3] XOR IN[2] OUT[5] = IN[4] XOR IN[3] XOR IN[2] XOR IN[1] OUT[4] = IN[] XOR IN[2] XOR IN[1] XOR IN[0]

Fig. 8B